МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«20» 02 2026 г.

КИБЕРФИЗИЧЕСКИЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Методические указания по лабораторным работам для студентов направления подготовки «Информатика и вычислительная техника»

Курск 2026

УДК 004.021

Составитель Д.О. Бобынцев

Рецензент: к.т.н., доцент Конаныхина Т.Н.

Киберфизические системы и технологии: методические указания к лабораторным работам / Юго-Зап. гос. ун-т; сост.: Д.О. Бобынцев. Курск, 2026. – 65 с.

Содержит методические указания по практическим и лабораторным занятиям дисциплины «Киберфизические системы и технологии». Даны теоретические материалы, описание порядка выполнения работ, контрольные вопросы, список литературы. Предназначено для студентов направления подготовки «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать 20.02.26. Формат 60х84 1/16.
Усл.печ. л. 4,36. Уч.-изд. л. 3,42. Тираж 100 экз. Заказ. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

2. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

2.1. Лабораторная работа №1

ЗНАКОМСТВО С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ PYTHON. УСТАНОВКА ПРОГРАММНОЙ СРЕДЫ PYTHON, ПАКЕТОВ NUMPY, PANDAS, IPYTHON

Целью лабораторной работы является знакомство с языком Python. Установка программной среды для работы с этим языком. Настройка программного окружения для дальнейшей работы.

В результате выполнения лабораторных работ студенты должны знать:

- Процедуру установку программной среды Python на ПК с операционной системой Windows
- Процедуру установки дополнительных пакетов
- Методы работы с командной строкой Windows

Используемые программно-технические средства: персональная ЭВМ класса IBM PC AT стандартной конфигурации; операционная система Windows XP/Vista/7/8, среда разработки для Python

Теоретические сведения

1. Общие сведения о языке программирования Python

Python является высокоуровневым языком общего назначения, применяющийся, помимо прочего, для выполнения статистических расчетов, моделирования и анализа данных. Для этого есть несколько причин:

- Понятный и удобочитаемый синтаксис.
- Богатая стандартная библиотека (в особенности, функции работы со строками). Доступно множество

сторонних библиотек, предназначенных для выполнения научно-исследовательских работ.

- Распространенность и поддержка. Python используют большое число людей и организаций во всем мире, поэтому он развивается и хорошо документирован.
- Открытость. Язык является кросс-платформенным и пользоваться им можно совершенно бесплатно.

Интуитивно понятный синтаксис Python зачастую называют исполняемым псевдокодом. Установка Python по умолчанию уже включает высокоуровневые типы данных, такие как списки, кортежи, словари, наборы, последовательности и так далее, которые уже нет необходимости реализовывать пользователю. Эти типы данных высокого уровня делают простой реализацию абстрактных понятий. Python позволяет программировать в любом знакомом вам стиле: объектно-ориентированном, процедурном, функциональном и так далее.

В Python просто обрабатывать и манипулировать текстом, что делает его идеальным для обработки нечисловых данных. Есть ряд библиотек для использования Python для доступа к веб-страницам, а интуитивно понятные манипуляции с текстом позволяют легко извлекать данные из HTML-кода.

Язык программирования Python популярен и множество доступных примеров кода делает обучение ему простым и достаточно быстрым. Популярность означает, что есть множество модулей предназначенных для различных приложений.

Python является популярным языком программирования в научных, а также финансовых кругах. Ряд библиотек для научных вычислений, таких как PandasиNumPy позволяют выполнять операции над векторами и матрицами. Это также делает код еще более читаемым и позволяет писать код, который выглядит как выражения линейной алгебры. Кроме того, научные библиотеки PandasиNumPy скомпилированы, используя языки низкого уровня (C и Fortran), что делает вычисления при использовании этих инструментов значительно быстрее.

Python также имеет интерактивную оболочку, которая позволяет просматривать и проверять элементы разрабатываемой программы.

2. Преимущества и недостатки Python

Преимущества:

- Встроенная поддержка "длинной арифметики", комплексных чисел, списков, словарей, стеков, очередей и т.д.
- Кроссплатформенность
- Поддержка всех кодировок
- Большое количество библиотек на все случаи
- Интерпретируемый язык (можно менять код без перекомпиляции). Удобно писать служебные скрипты.
- Сборщик мусора позволяет не беспокоиться об освобождении памяти.
- Python сравнительно прост в изучении и позволяет выражать алгоритмы кратко и просто.

Недостатки:

- Низкая, в сравнении с компилируемыми языками, скорость выполнения (причины: интерпретация, динамическая типизация).
- Отсутствие библиотек для создания графических интерфейсов для Windows.

3. Сборщик пакетов pip

Сборка пакетов имеет решающее значение для успешной реализации проекта с открытым исходным кодом. Ключевой составляющей правильной сборки является управление версиями. Так как проект имеет открытый исходный код, вы можете захотеть опубликовать пакет, чтобы реализовать все преимущества, предоставляемые сообществом разработчиков программ с от-

крытым исходным кодом. В различных платформах или языках используются разные механизмы сборки пакетов, однако данная статья посвящена именно Python и его экосистеме сборки пакетов. В статье обсуждаются механизмы сборки пакетов, обеспечивающие основу для развития, а также приводится достаточно примеров, на основании которых можно сразу же приступать к действию.

Помимо того, что это просто хороший тон, существуют еще три практические причины для поставки программного обеспечения в виде пакетов:

- простота использования;
- стабильность (при управлении версиями);
- распространение.

При выборе приложения простота его установки учитывается пользователями далеко не в последнюю очередь, поэтому следует максимально упростить этот процесс. Сборка пакетов позволяет сделать программное обеспечение более доступным и простым в установке. Если установка не сложная, значит, пользователям будет проще приступить к работе с вашим программным обеспечением. Повысить доступность своего пакета при его публикации в каталоге пакетов Python (PythonPackageIndex — PyPI) можно посредством таких утилит как pip или easy_install

Управление версиями обеспечивает лучшую устойчивость при внесении изменений в программное обеспечение в будущем, в результате которых могут возникнуть конфликты в интерфейсах. В итоге ваши пользователи точно знают, что они получают, и им легче отслеживать различия в разных версиях. К тому же, разработчикам проектов точно известно, над чем они работают.

Наиболее популярным методом публикации пакетов на PyPI (или на вашем собственном дистрибутивном сервере) является создание дистрибутива исходного кода для его свободной загрузки. Дистрибутив исходного кода — это стандартный способ сборки кода вашего проекта в качестве распространяемого

модуля. Можно также создавать бинарные дистрибутивы, но для целей открытого использования имеет смысл также распространять свой код. Создавая дистрибутивы исходного кода, вы облегчаете пользователям процесс нахождения программного обеспечения в сети Интернет с помощью автоматизированных средств, а также его загрузки и установки. Этот процесс помогает не только в разработке на локальных системах, но и в развертывании вашего программного обеспечения.

Таким образом, облегчая для пользователей процесс интегрирования и установки вашего программного обеспечения, применяя эффективное управление версиями, обеспечивающее надежные методы закрепления за версиями, и публикуя свои пакеты для их более широкого распространения, вы увеличиваете шансы на успех своего проекта и на получение более широкого признания. Более широкое распространение вашего кода может привести к увеличению числа участников, вносящих вклад в развитие кода — т. е. к достижению той цели, которую ставит перед собой каждый разработчик программ с открытым исходным кодом.

4. Пакет NumPy

NumPy является основным пакетом для научных вычислений в Python. NumPy является расширением языка программирования Python, добавляющим поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для работы с этими массивами. Предшественник NumPy, пакет Numeric, был первоначально создан Джимом Хаганином при участии ряда других разработчиков. В 2005 году ТрэвисОлифант создал NumPy путем включения функций конкурирующего пакета Numarray в Numeric, произведя при этом обширные изменения.

5. Пакет Pandas

Pandas — это пакет *Python,* предназначенный для обеспечения быстрыми, гибкими, и выразительными структурами данных, упрощающими работу с «относительными» или «помечеными» данными простым и интуитивно понятным способом. *pandas* стремится стать основным высокоуровневым строительным блоком для проведения в *Python* практического анализа данных, полученных из реального мира. Кроме того, этот пакет претендует стать самым мощным и гибким *open-source* инструментом для анализа/обработки данных, доступным в любом языке программирования.

Pandas хорошо подходит для работы с различными типами данных:

- Табличные данные со столбцами различных типов, как в таблицах *SQL* или *Excel*.
- Упорядоченными и неупорядоченными данными (не обязательно с постоянной частотой) временных рядов.
- Произвольными матричными данными (однородными или разнородными) с помеченными строками и столбцами.

  Любыми другими формами наборов данных наблюдений, либо статистических данных. Данные на самом деле не требуют обязательного наличия метки для того, чтобы быть помещенными в структуру данных *pandas*.

6. Пакет IPython

IPython является командной оболочкой для интерактивных вычислений на нескольких языках программирования, первоначально разработанной для языка программирования Python.

IPython позволяет расширить возможности представления, добавляет синтаксис оболочке, автодополнение и обшир-

ную историю команд. IPython в настоящее время предоставляет следующие возможности:

- Мощные интерактивные оболочки (терминального типа и основанную на Qt).
- Браузерный редактор с поддержкой кода, текста, математических выражений, встроенных графиков и других возможностей представления.
- Поддерживает интерактивную визуализацию данных и использование инструментов GUI.
- Гибкие, встраиваемые интерпретаторы для работы в собственных проектах.
- Простые в использовании, высокопроизводительные инструменты для параллельных вычислений.

Практическая часть

1. Установка Python

Для установки Python необходимо произвести следующие шаги:

1) перейдите на сайт http://www.python.org/
2) на вкладке «Downloads» выберете свою операционную систему
3) выберете и скачайте интересующую вас версию(Python 3.4.+).
4) запустите скаченный установочный файл и следуйте инструкциям по установке

После того как все будет установлено, можно начинать работу с Python. Есть возможность работы в двух режимах — в командной строке и через IDLE (Shell оболочка для Python). Выбор режима работы показан на рис. 2.1.

Рис. 2.1. Выбор режима работы с Python

Пример работы в командной строке приведен на рис. 2.2:

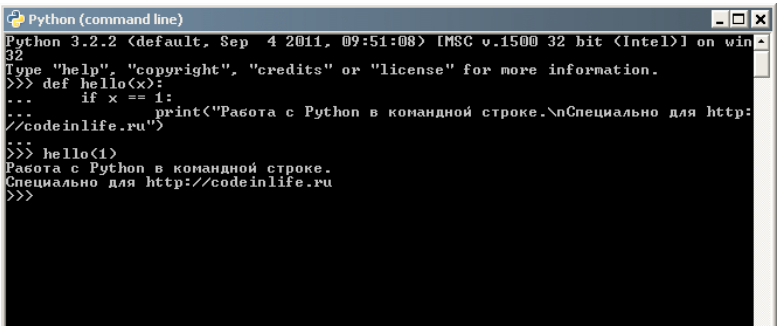

Рис. 2.2. Пример работы в командной строке

Пример работы через IDLE (рис. 2.3):

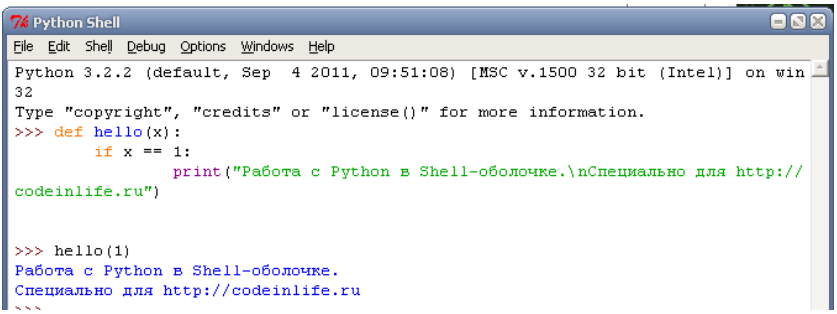

Рис. 2.3. Пример работы в оболочке IDLE

Чтобы узнать версию установленного пакета введите следующие команды:

```
importsys
sys.version
```

Данная функция возвращает версию установленного пакета Python (рис. 2.4):

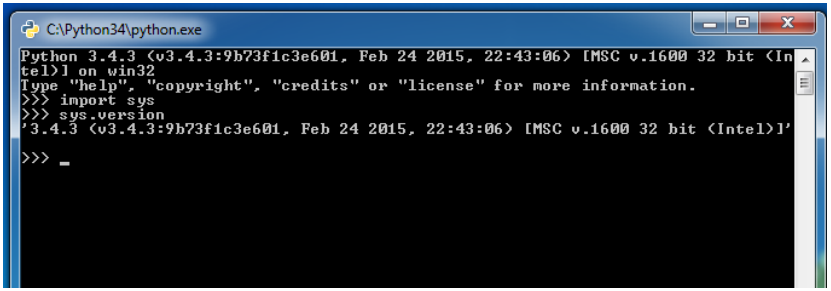

Рис. 2.4. Версия установленного пакета Python

2. Установка сборщика пакетов pip

Для начала установки пакетов нужно установить установщик пакетов "pip"

1) Скачайте get-pip.py, который находится по ссылке https://bootstrap.pypa.io/get-pip.py. Нажмите правой кнопкой мыши на тексте. Выберите опцию "Сохранить как...". Сохраните файл с тем же именем с расширением ".py"
2) Выполните скаченный файл, в среде Python, при этом будет установлен Pip (данный инструмент необходим для легкой установки модулей)
3) Откройте командную строку (рис. 2.5) и перейдите в папку %путь до каталога Python%\Scripts\ (командой «cd»).
4) Из данной строки можно запускать установку пакетов для Python с помощью команды:

```
pip install %name_of_package"
```

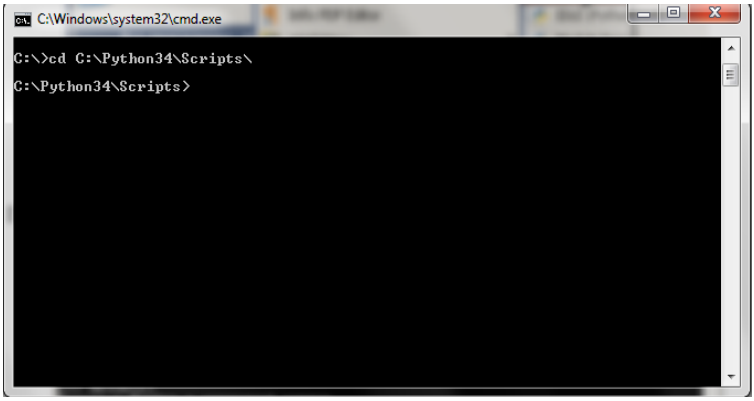

Рис. 2.5. Переход в командной строке Windows в папку со скриптами Python

3. Установка пакетов Pandas, Numpy, IPython

Для установки пакетов Python, необходимых для дальнейших лабораторных работ:

1) Запустите командную строку Windows от имени Администратора, перейдите в папку с скриптами Python(C:\Python34\Scripts)
2) Введите команды для установки соответствующих пакетов:

```
pip install pandas
pip install numpy
pipinstallipython
```

4. Установка пакета Anaconda (Miniconda)

Пакет Anaconda компании Continuumanalytics – это свободно распространяемый дистрибутив интерпретатора Python для обработки больших данных, аналитики и научных расчетов. Продукт состоит интерпретатора языка Python,менеджера пакетов conda, при помощи которого можно загружать, обновлять и удалять библиотеки python, управлять зависимостями, и нескольких сотен устанавливаемых библиотек.

Для скачивания последней версии пакета, необходимо перейти по ссылке http://continuum.io/downloads#py34, выбрать ссылку с нужной версией операционной системы, после чего начнется скачивание инсталлятора.

После установки дистрибутива, необходимо запустить командную строку или терминал и ввести название нужной программы, например: python qtconsole.

2.2. Лабораторная работа №2

РАБОТА С CSV-ФАЙЛАМИ

Цель работы: познакомить студента с форматом хранения данных .csv. Разобрать функции Pythonдля работы с данными такого формата.

В результате выполнения лабораторных работ студенты должны знать:

- Формат файлов.csv.
- Научиться считывать данные из файлов такого формата и записывать в них. Выводить данные пользователю.
- Построение графиков данных из формата .csv

Используемые программно-технические средства: персональная ЭВМ класса IBM PC AT стандартной конфигурации; операционная система Windows XP/Vista/7/8, среда разработки для Python

Теоретические сведения

1. Работа с CSV файлами

Файл в формате CSV (comma-separatedvalues - значения, разделенные запятыми) - текстовая структура данных, предназначенная для переноса табличной информации между приложениями (электронными таблицами, СУБД, адресными книгами и т.п.). Каждая строка файла представляет собой запись, каждое поле которой отделено от другого символом-разделителем (чаще всего - запятой).

Внутри файл выглядит примерно так:

```
name,number,text
a,1,something here
b,2,"one, two, three"
c,3,"no commas here"
```

Для работы с CSV файлами необходимо подключить библиотеку pandas. Она предназначена для всего анализа данных, исключением является небольшой "кусок" в разделе представления данных.

Для вывода полученных данных в таблицу, также необходимо подключить библиотеку Matplotlib.

Практическая часть

1. Создание данных

Первым шагом в выполнении лабораторной работы будет подключение всех необходимых библиотек.

Ввод:

```
# Подключение отдельной функции из библиотеки
# from (имя_библиотеки) import (имя_функции)
from pandas import DataFrame, read_csv

# Общий синтаксис подключения библиотек без функций:
# import (имя_библиотеки) as (псевдоним)
import matplotlib.pyplot as plt
import pandas as pd
import sys # Строка нужна для того, чтобы определить
версию Python

# Подключение inline-режима для построения графиков
%matplotlibinline

# Вывод на экран версии python
print('Python version',sys.version)
# Вывод на экран версии pandas
print('Pandas version ',pd.__version__)
```

Вывод:

```
Python version 3.4.3 |Anaconda 2.1.0 (64-bit)| (de-
fault, Jun  1 2015, 12:37:52) [MSC v.1500 64 bit
(AMD64)]
Pandasversion 0.16.1
```

После подключения библиотек, необходимо создать набор данных. К примеру, он будет состоять из пяти имен и количества рожденных детей с таким именем в России в 2015 году.

Ввод:

```
# Начальный набор данных
names = ['Саша', 'Маша', 'Гоша', 'Тоша', 'Антоша']
births = [968, 155, 77, 578, 973]
```

Для объединения этих двух списков воспользуемся функцией zip().

Ввод:

```
BabyDataSet = list(zip(names,births))
BabyDataSet
```

Вывод:

```
[('Саша', 968), ('Маша', 155), ('Гоша', 77), ('Тоша',
578), ('Антоша', 973)]
```

После создания набора данных, экспортируем их в CSV файл.

Ввод:

```
df = pd.DataFrame(data = BabyDataSet, columns=['Names',
'Births'])
df
```

DF это объект DataFrame, т.е. проиндексированный многомерный массив значений. По аналогии схож с таблицей SQL или таблицей Excel. Ниже представлено его содержимое.

Вывод:

	Names	Births
0	Саша	968
1	Маша	155

	Names	Births
2	Гоша	77
3	Тоша	578
4	Антоша	73

Экспортируем полученную таблицу в CSV файл и назовем его births2015.csv. Функция to_csv позволит экспортировать наши данные в файл и сохранит его. Файл будет сохранен в корне каталога компилятора Python.

Если назначим параметрам index (индекс) и header (заголовок) ложные значения —индексы и заголовки не сохранятся в файл.

Ввод:

```
df.to_csv ("births2015.csv", index = False, header =
False, encoding="utf8")
```

2. Получение данных

Чтобы вновь воспользоваться сохраненными данными в CSV файлевоспользуемся функцией "read_csv". Более подробно почитать про эту функцию можно набрав в интерпретаторе команду read_csv.

Передадим в функцию read_csv(), через переменную расположение CSV файла.

Ввод:

```
Location = r"births2015.csv"
df = pd.read_csv(Location)
```

Примечание: Путь зависит, от того, где сохранен файл.

Символ r в начале строки предупреждает компилятор о том, что специальные символы такие как «/»всего лишь текст.

Ввод:

```
df
```

Вывод:

	Саша	968
0	Маша	155
1	Гоша	77
2	Тоша	578
3	Антоша	973

После вывода данных, прочитанных из csv файла, на экран, появится проблема в правильности наименования заголовков, т.к. в конкретном примере у таблицы их быть не должно. Чтобы это исправить, передадим в параметр header функции read_csv значение None, т.е. явным образом убираем информацию в заголовках.

Ввод:

```
df = pd.read_csv(Location, header=None)
df
```

Вывод:

	0	1
0	Саша	968
1	Маша	155
2	Гоша	77
3	Тоша	578
4	Антоша	973

Для того чтобы называть столбцы конкретными именами, добавим их имена в параметр names.

Ввод:

```
df = pd.read_csv(Location,
names=['Names','Births'])
df
```

Вывод:

	Names	Births
0	Саша	968
1	Маша	155
2	Гоша	77
3	Тоша	578
4	Антоша	973

Можно предположить, что индексы [0, 1, 2, 3, 4] подобны строками в Excel, однако для библиотеки pandas они представляют собой часть индекса выбранного набора данных. Эти индексы нельзя использовать в SQL запросах в виде первичного ключа, потому что индексы могут дублироваться.

[Names, Births] это названия столбцов, аналогично заголовкам электронных таблиц в Excel.

После загрузки набора данных в память компьютера, можно удалить исходный CSV файл.

Ввод:

```
importos
os.remove(Location)
```

3. Подготовка данных

На предыдущем этапе, мы получили набор с данными, состоящего из имен и количества рожденных детей с этими именами. Рассмотрим эти данные более подробно.

В колонке Names записаны имена в алфавитно-цифровом виде, т.е. там могут быть различные символы. Колонка Births должна содержать только целые числа. Необходимо проверить все ли данные из этой колонки целые числа или нет.

Ввод:

```
# Проверка типа данных столбцов
df.dtypes
```

Вывод:

```
Names     object
Births     int64
dtype: object
```

Ввод:

```
# Проверка типа данных столбца Births
df.Births.dtype
```

Вывод:

```
dtype('int64')
```

Так как столбец Births имеет тип данных int64, тип данных изменять не нужно и это удовлетворяет поставленному условию.

4. Анализ данных

Для того чтобы найти самое популярное имя среди детей, рожденных в 2015 году, необходимо определить наибольший показатель Births. Это можно сделать двумя способами:

- Отсортировать набор данных по убыванию значений в выбранном поле и выбрать первуюстроку;
- Воспользоваться процедурой max(), возвращающей максимальное значение в наборе, над нужным полем.

Ввод:

```
# Метод 1:
Sorted = df.sort(['Births'], ascending=False)
Sorted.head(1)
```

Вывод:

	Names	Births
4	Антоша	973

Ввод:

```
# Метод2:
df['Births'].max()
```

Вывод:

```
973
```

5. Представление данных

По значениям столбца Births можно построить график, чтобы графически было видно самую высокую точку рождаемости. В сочетании с таблицей пользователь будет точно знать, что «Антоша» самое популярное имя в DataFrame (рис. 2.6).

Процедура plot() пакета pandas позволяет легко и безболезненно манипулировать данными таблицы.

На предыдущем шаге мы получили максимальное значение поля. Теперь, чтобы найти соответствующее этому значению имя необходимо провести выборку, но для начала разберем некоторые полезные части кода:

- df['Names'] –список всех имен;
- df['Births'] – список значений в поле «количество рождений»;
- df['Births'].max() – это максимальное значение находящееся в поле «количество рождений»;
- [df['Births'] == df[‘Births'].max()] – эквивалентно запросу: найти все записи в поле «количество рождений», где его значение равно максимальному;
- df['Names'][df['Births'] == df['Births'].max()] –запрос вернет все записи в поле Names, где значение в поле Births равно максимальномузначениюввыбранномнабореданных.

Также, можно получить значение имени из созданного на предыдущем шаге наборе «Sorted»:

```
Sorted['Names'].head(1).values[0]
```

Используемая далее функция str() преобразует входящий объект в строку.

Ввод:

```
# Установка шрифта, поддерживающего кириллицу
plt.rc('font', family='Arial')
# Создание графика
df['Births'].plot()
# Максимальное значение в наборе данных
MaxValue = df['Births'].max()
# Имя, связанное с максимальным значением
MaxName = df['Names'][df['Births'] ==
df['Births'].max()].values
# Текст отображающийся на графике
Text = str(MaxValue) + " - " + MaxName
# Добавление текста на график
plt.annotate(Text, xy=(1, MaxValue), xytext=(8, 0),
```

```
xycoords=('axes fraction', 'data'), textcoords='offset
points')

print "Самоепопулярноеимя:"
df[df['Births'] == df['Births'].max()]
```

Вывод:

```
Самоепопулярноеимя:
```

	Names	Births
4	Антоша	973

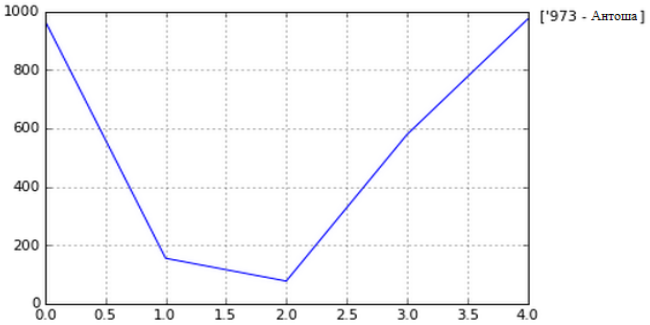

Рис. 2.6. График популярности имён

6. Чтение данных из CSV файла

Чтобы получить данные из файла CSV необходимо использоваться функцией read_csv. В качестве источника данных для анализа возьмем набор данных велосипедных дорожек Монреаля за 2012 год в виде csv (загрузить по ссылке: https://github.com/jvns/pandas-cookbook/blob/v0.1/data/bikes.csv,

либо взять у преподавателя). Исходные данные представлены в виде списка велосипедных дорожек.

Ввод:

```
broken_df = pd.read_csv('2012.csv')
# Просмотр первых трех строк таблицы
broken_df[:3]
```

Вывод:

	��1; .���2���.��.������.��� ������ ���1
0	01/01/2012;35;;0;38;51;26;10;16;
1	02/01/2012;83;;1;68;153;53;6;43;
2	03/01/2012;135;;2;104;248;89;3;58;

Как видно из полученного результата, данные из csv файла, не всегда могут быть корректно прочитаны без дополнительных настроек. Однако указание правильных значений атрибутов функции read_csv позволяет это исправить. Для этого необходимо проделать ряд изменений:

- установить разделителем полей символ «точка с запятой»;
- поменять кодировку на *latin1* (изначально *utf-8*);
- разобрать дату из поля «Date»;
- привести дату к виду год-месяц-число;
- назначить поле с датой в качестве индекса.

Ввод:

```
fixed_df = pd.read_csv('../data/bikes.csv', sep=';',
encoding='latin1', parse_dates=['Date'], dayfirst=True,
index_col='Date')
fixed_df[:3]
```

Вывод:

Дата	Берри 1	Бербиф (данные недо-ступны)	Кот-Сент-Катрин	Мезоннев 1	Мезоннев 2	Парк	Пьер-Дюпюи	Рэйчел 1	Санкт-Урбан (данные недо-ступны)
2012-01-01	35	NaN	0	38	51	26	10	16	NaN
2012-01-02	83	NaN	1	68	153	53	6	43	NaN
2012-01-03	135	NaN	2	104	248	89	3	58	NaN

Выборка столбца

При считывании данных с файла CSV получаем объект DataFrame, который состоит из строк и столбцов. И из этого объекта можно осуществлять выборку столбцов.

Ввод:

```
fixed_df['Berri 1']
```

Вывод:

```
Date:
2012-01-01     35
2012-01-02     83
2012-01-03    135
2012-01-04    144
2012-01-05    197
...
2012-10-31    2634
2012-11-01    2405
2012-11-02    1582
2012-11-03     844
2012-11-04     966
2012-11-05    2247
Name: Berri 1, Length: 310, dtype: int64
```

Построение графика

Для построения графика выбранного столбца воспользуемся функцией plot() (см. рис. 2.7).

Ввод:

```
fixed_df['Berri 1'].plot()
```

Вывод:

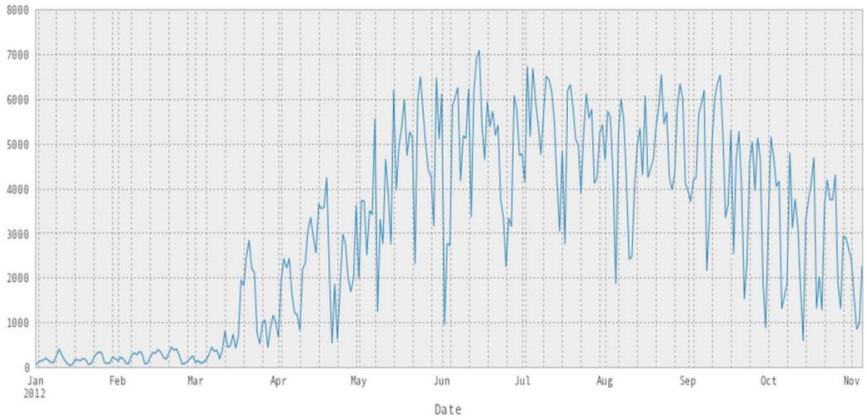

Рис. 2.7. График осадков, составленный с помощью функции plot()

Также можно представить в виде графика все столбцы набора (рис. 2.8).

Ввод:

```
fixed_df.plot(figsize=(15, 10))
```

Вывод:

```
<matplotlib.axes.AxesSubplot at 0x3fc2110>
```

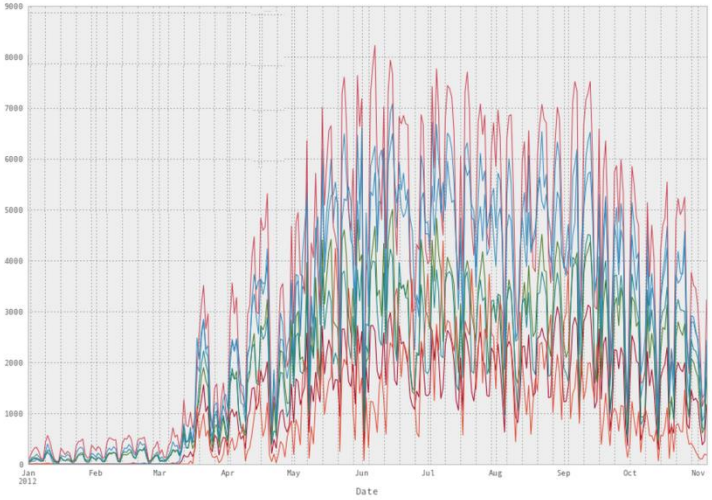

Рис. 2.8. График, составленный по всем столбцам с помощью функции plot()

2.3. Лабораторная работа №3

РАБОТА С ФАЙЛАМИ ТЕКСТОВОГО ФОРМАТА

Цель работы: изучение основных функций при работе с текстовыми файлами и получения практических навыков.

В результате выполнения данной лабораторной работы студент:

- Узнает о наиболее популярных используемых форматах текстовых файлов
- Научится создавать данные, извлекать, вводить в текстовые файлы, используя программные средства Python
- Формализовывать данные и приводить их к единому формату с помощью методов пакетов Python

Используемые программно-технические средства: персональная ЭВМ класса IBM PC AT стандартной конфигурации; операционная система Windows XP/Vista/7/8, среда разработки для Python

Теоретические сведения

Текстовый файл представляет из себя последовательность символов (в основном печатных знаков, принадлежащих тому или иному набору символов). Эти символы обычно сгруппированы в строки (англ. *lines, rows*). В современных системах строки разделяются разделителями строк, в прошлом же применялось хранение строк в виде записей постоянной или переменной длины. Иногда конец текстового файла (особенно если в файловой системе не хранится информация о размере файла) также отмечается одним или более специальными знаками, известными как маркеры конца файла.

Текстовой файл может содержать как форматированный, так и неформатированный текст.

На данный момент наиболее распространены следующие форматы текстовых файлов:

TXT («простой текстовый») Наиболее универсальный формат. Активно используется по сей день. Сохраняет текст без форматирования, в текст вставляются только управляющие символы конца абзаца. Так как текст хранится в виде последовательности символов, то размер файла в байтах равен числу символов плюс непечатаемые символы (знак пробела, табуляции, знак конца абзаца и другие - их ещё называют знаками форматирования). За счёт этого достигается малый размер файла. Однако возможности по форматированию подобных документов сильно ограничены. Применяют этот формат для хранения документов, которые должны быть прочитаны в приложениях, работающих в различных операционных системах.

RTF («RichTextFormat» - «формат обогащённого текста») Свободный межплатформенный формат хранения размеченных текстовых документов, созданный Microsoft в 1987 году. Ныне он широко распространён, поэтому большинство современных текстовых редакторов его поддерживают. Создав RTF на платформе Windows, он прекрасно будет читаться и редактироваться на других платформах (Apple, Linux и другие). Стандарт дефакто в полиграфии. Формат RTF позволяет производить и сохранять достаточно сложное форматирование, вставлять сноски, колонтитулы, рисунки, таблицы и формулы, хотя в этом он все же уступает формату DOC. Уступает он DOC и в объёме файлов: сложные документы более компактно хранятся в DOC-файлах (простые - наоборот). Однако RTF выигрывает спор с DOC в отношении безопасности, так как не использует макросы.

DOC (от англ. «document») Поначалу это расширение использовалось для обозначения простых текстовых файлов без форматирования, однако в начале 90-х Microsoft фактически его «приватизировала». Поэтому сейчас DOC ассоциируется только с продуктами этой компании. Этот формат обеспечивает большие возможности по форматированию текста (включены сцена-

рии, макросы). За счёт этого ухудшилась совместимость с текстовыми редакторами сторонних разработчиков. Однако при включении в документ различных графических элементов и изображений DOC выигрывает в размере и обеспечивает большую совместимость. В отличие от TXT и RTF DOC является бинарным форматом, что делает его нечитабельным в простых текстовых редакторах. К примеру, «блокнот» может просматривать некоторые RTF-файлы. Популярен наравне с RTF.

DOCX. С появлением Office 2007 компания Microsoft перешла на новые форматы, базирующиеся на OfficeOpen XML (визуально отличаются тем, что к расширениям добавлена буква «x» на конце). Формат представляет собой zip-архив, содержащий текст в виде XML, графику и другие данные. Для уменьшения размера файла используется ZiP-компрессия. Документы обратно совместимы с Office 2000/XP/2003, только если установленMicrosoftOfficeCompatibilityPack. Так же документы в формате DOCX совместимы с последними версиями OoenOfficeWriter и LibraOfficeWriter.

ODT/ODF («OpenDocumentFormat») ODF - общее наименование открытого формата документов для офисных приложений (текст, таблицы, рисунки, базы данных, презентации). Текстовые данные хранятся в файлах с расширением ODT. Стандарт был разработан индустриальным сообществом OASIS и основан на XML-формате. 1 мая 2006 года принят как международный стандарт ISO/IEC 26300. ODF доступен для всех и может быть использован без ограничений. Бесплатная альтернатива закрытым форматам Microsoft. Документы могут читаться офисным пакетом MicrosoftOffice версии 2007 и выше.

HTML (от англ. HypertextMarkupLanguage - «язык разметки гипертекста») Стандартный язык разметки документов в интернете (расширение .htm/html). Веб-страницы создаются при помощи языка HTML (или XHTML).

PDF (PortableDocumentFormat - «переносимый формат документов») Кроссплатформенный формат электронных доку-

ментов, созданный фирмой AdobeSystems с использованием ряда возможностей языка PostScript. В первую очередь предназначен для представления в электронном виде полиграфической продукции. Традиционным способом создания PDF-документов является следующий: документ как таковой готовится в своей программе, а затем экспортируется в PDF. Некоторые программы имеют возможность для прямого экспорта (без использования виртуального принтера). Например, OpenOffice.org. Стандарт де-факто для большинства документации.

DjVu («дежавю») Технология сжатия изображений с потерями, разработанная специально для хранения сканированных документов - книг, журналов, рукописей и пр., где наличие формул, схем, рисунков и рукописных символов делает чрезвычайно трудоёмким их полноценное распознавание. Также является эффективным решением, если необходимо передать все нюансы оформления, например, исторических документов. Суть технологии DjVu заключается в автоматическом разбиении изображения на несколько участков (например, текст, логотип фирмы и растровая фотография), для каждого из которых выбирается оптимальный алгоритм сжатия. Кроме того, DjVu-файл может содержать встроенное интерактивное оглавление и активные области - ссылки, что позволяет реализовывать удобную навигацию. Дает выигрыш в размере файла по сравнению с GIF-форматом в среднем в полтора-два десятка раз.

XML-форматы (extensibleMarkupLanguage - «расширяемый язык разметки») Существует довольно много текстовых форматов, созданных для одного конкретного устройства или программы. Например, электронные книги. К ним можно отнести Rocket e-book (.rb), MicrosoftReader (.lit), PalmDoc, MobiPocket (.pro) и т.д. Как правило, все они созданы с помощью языка XML. FictionBook (FB2) - формат представления электронных версий книг в виде XML-документов. Стандарт призван обеспечить совместимость с любыми устройствами и форматами. XML позволяет легко создавать документы, готовые

к непосредственному использованию и программной обработке (конвертации, хранению, управлению) в любой среде. Документы, обычно имеющие расширение.fb2, могут содержать структурную разметку основных элементов текста (главы, заголовки, цитаты, врезки), некоторое количество информации о книге, а также могут содержать вложения с двоичными файлами, в которых могут храниться иллюстрации или обложка. Основное преимущество FictionBook(.fb2) - возможность без труда создавать (в том числе и автоматически) книги в этом формате из файлов всех популярных текстовых форматов (*.txt, *.doc, *.rtf. *.html и пр.). ElectronicPublication (ePub) — открытый формат электронных версий книг, разработанный Международным форумом по цифровым публикациям. Файлы в этом формате имеют расширение .epub. Формат позволяет издателям производить и распространять цифровую публикацию в одном файле, обеспечивая совместимость между программным и аппаратным обеспечением, необходимым для воспроизведения цифровых книг и других публикаций с плавающей вёрсткой.

Практическая часть

1. Работа с текстовыми файлами

Для формирования шаблонных данных, необходимо подключить библиотеку NumPy.

Ввод:

```
# Подключение необходимых библиотек
Importpandasaspd
Fromnumpyimportrandom
Importmatplotlib.pyplot as plt
import sys

%matplotlib inline
plt.rc('font', family='Arial')

print("Python version",sys.version)
print("Pandas version",pd.__version__)
```

Вывод:

```
Python version 3.4.3 |Continuum Analytics, Inc.| (de-
fault, Jan  6 2015, 12:08:17) [MSC v.1600 32 bit (In-
tel)]
Pandasversion 0.16.1
```

2. Создание данных

Набор данных будет состоять из 1000 имен и количества рожденных детей с таким именем в России в 2015 году. Мы будем дублировать имена, чтобы одно и то же имя можно было встретить несколько раз.

Ввод:

```
# Начальный набор данных
names = ['Саша', 'Маша', 'Гоша', 'Тоша', 'Антоша']
```

Для того чтобы создать 1000 имен сделаем следующее:

Возьмем случайное число между 0 и 4, при помощи функций seed, randint, len, range и zip().

seed(500) – создает начальное заполнение генератора случайных чисел.

randint(low=0,high=len(names)) - генерируется случайное целое число от 0 до 4;

names[n] –получаем значение по индексу n из набора names;

foriinrange(n) – цикл по n элементам с шагом 1;

random_names[:n]- созданная нами функция, возвращающая n случайных имен.

Ввод:

```
random.seed(500)
random_names =
[names[random.randint(low=0,high=len(names))] for i in
range(1000)]
```

```
# Вывод на экран первых 10 имен
random_names[:10]
```

Вывод:

```
['Гоша',
 'Маша',
 'Маша',
 'Саша',
 'Маша',
 'Маша',
 'Маша',
 'Гоша',
 'Гоша',
 'Гоша']
```

Далее, создадим набор из 10 случайных чисел от 0 до 1000.

Ввод:

```
births = [random.randint(low=0,high=1000) for i in
range(1000)]
births[:10]
```

Вывод:

```
[968, 155, 77, 578, 973, 124, 155, 403, 199, 191]
```

Объединим наборы с именами и числами с помощью функции zip().

Ввод:

```
BabyDataSet = list(zip(random_names,births))
BabyDataSet[:10]
```

Вывод:

```
[('Гоша', 968),
 ('Маша', 155),
 ('Маша', 77),
 ('Саша', 578),
```

```
 ('Маша', 973),
 ('Маша', 124),
 ('Маша', 155),
 ('Гоша', 403),
 ('Гоша', 199),
 ('Гоша', 191)]
```

Итак, набор данных готов. Теперь, используя библиотеку pandas, экспортируем эти данные в текстовый файл. Для начала, создадим объект DataFrame, который будет содержать созданный нами набор данных.

Ввод:

```
df = pd.DataFrame(data = BabyDataSet, columns=['Names',
'Births'])
df[:10]
```

Вывод:

	Names	Births
0	Гоша	968
1	Маша	155
2	Маша	77
3	Саша	578
4	Маша	973
5	Маша	124
6	Маша	155
7	Гоша	403
8	Гоша	199
9	Гоша	191

Экспортируем полученную таблицу в текстовый файл и назовем его births2015.txt. Функция to_csv позволит экспортировать наши данные в файл и сохранит его в каталоге с выполняющим скриптом.

Если назначим параметрам index (индекс) и header (заголовок) значение *False*—индексы и заголовки не сохранятся в файл.

Ввод:

```
df.to_csv('births2015.txt',index=False,header=False,
encoding="utf8")
```

3. Получение данных

Для того, чтобы прочитать сохраненные на предыдущем шаге данные, воспользуемся функцией read_csv. Передадим ей расположение текстового файла.

Ввод:

```
Location = r'births2015.txt'
df = pd.read_csv(Location)
df.info()
```

Вывод:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 999 entries, 0 to 998
Data columns (total 2 columns):
Гоша999 non-null object
968     999 non-null int64
dtypes: int64(1), object(1)
memory usage: 19.5+ KB
```

Расшифровка:

- Имеется 999 записей с 2 полями в каждой;
- В поле «Гоша» содержатся 999 значений (мы разберемся с этим позже);
- В поле «968» содержатся 999 значений;

- Одно из полей является числовым, другое - не числовое.

Чтобы отобразить содержимое набора данных, необходимо воспользоваться функцией head(), которая по умолчанию возвращает первые пять записей в наборе. Передаваемый в скобках параметр, позволит получить указанное количество значений.

Ввод:

```
df.head()
```

Вывод:

	Гоша	968
0	Маша	155
1	Маша	77
2	Саша	578
3	Маша	973
4	Маша	124

Очевидно, что функция read_csv определила первую строку данных, как заголовки набора. Это неправильно, и чтобы это исправить, нужно передать в параметр header функции read_csv значение None.

Ввод:

```
df = pd.read_csv(Location, header=None)
df.info()
```

Вывод:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 0 to 999
Data columns (total 2 columns):
```

```
0    1000 non-null object
1    1000 non-null int64
dtypes: int64(1), object(1)
memory usage: 19.5+ KB
```

Расшифровка:

- В наборе содержатся1000 записей;
- В поле «0» содержатся 1000 значений;
- В поле «1» содержатся 1000 значений;
- Типы данных в полях: числовой и объектный.

Выведем последние пять записей таблицы при помощи функции tail().

Ввод:

```
df.tail()
```

Вывод:

	0	1
0	Тоша	151
1	Маша	511
2	Тоша	756
3	Маша	294
4	Тоша	152

Так же можно назвать столбцы конкретными именами добавив их имена в параметр names.

Ввод:

```
df = pd.read_csv(Location, names=['Names','Births'])
df.head(5)
```

Вывод:

	Names	Births
0	Тоша	151
1	Саша	511
2	Тоша	756
3	Саша	294
4	Тоша	152

Для удаления текстового файла можно воспользоваться процедурой remove() из библиотеки os.

Ввод:

```
Importos
os.remove(Location)
```

4. Выборка данных

Найти количество уникальных значений в поле Names можно двумя способами.

Первый метод заключается в использовании функции unique().

Ввод:

```
df['Names'].unique()
```

Вывод:

```
array(['Гоша', 'Маша', 'Саша', 'Тоша', 'Антоша'],
dtype=object)
```

Для вывода значений на экран можно пройтись циклом по полученному массиву и выводить значения с помощью конструкции print:

```
# Вывод на экран уникальных значений
Forxindf['Names'].unique():
print(x)
```

Вывод:

```
Гоша
Маша
Саша
Тоша
Антоша
```

Метод второй
Ввод:

```
print(df['Names'].describe())
```

Вывод:

```
count     1000
unique       5
topСаша
freq       206
Name: Names, dtype: object
```

Для того чтобы сгруппировать количество рожденных детей по имени, можно воспользоваться функцией groupby().
Ввод:

```
# Создание объекта группировки по полю Names
name = df.groupby('Names')
# Применение функцию суммирования к сгруппированному
набору
df = name.sum()
df
```

Вывод:

Names	Births
Антоша	102319
Гоша	99438
Маша	97826
Саша	106817
Тоша	90705

5. Анализ данных

Чтобы найти самое популярное имя среди детей, рожденных в 2015 году, необходимо определить наибольший показатель рождаемости. Для этого используем процедуру max().

Ввод:

```
# Метод 1:
Sorted = df.sort(['Births'], ascending=False)
Sorted.head(1)
```

Вывод:

Names	**Births**
Саша	**106817**

Ввод:

```
# Метод 2:
df['Births'].max()
```

Вывод:

```
106817
```

6. Представление данных

По значениям столбца Births можно построить график при помощи функции plot(), чтобы графически было видно самую высокую точку рождаемости (рис. 2.9). По графику и отсортированной таблице можно определить, что Саша - самое популярное имя в наборе.

Ввод:

```
# Создание графика
df['Births'].plot(kind='bar')

print "Самоепопулярноеимя:"
df.sort(columns='Births', ascending=False)
```

Вывод:

```
Самоепопулярноеимя:
```

Names	Births
Саша	106817
Антоша	102319
Гоша	99438
Маша	97826
Тоша	90705

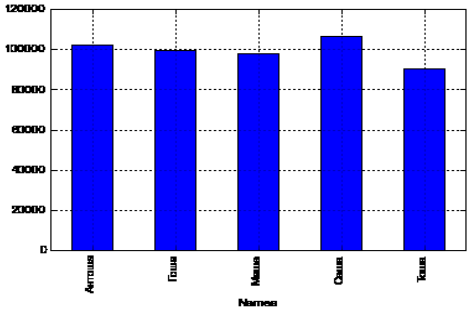

Рис. 2.9. График, отображающий наибольший показатель рождаемости

7. Сведения о наборе данных

Для просмотра сведений о наборе данных, можно воспользоваться функцией info() объекта DataFrame. Сведения включают в себя список полей, и количество непустых значений в каждом из них.

Ввод:

```
complaints = pd.read_csv('311-service-requests.csv')
complaints.info()
```

Вывод:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 111069 entries, 0 to 111068
Data columns (total 52 columns):
Unique Key                      111069  non-null values
Created Date                    111069  non-null values
Closed Date                     60270   non-null values
Agency                          111069  non-null values
.....................................................
Latitude                        98143  non-null values
Longitude                       98143  non-null values
Location                        98143  non-null values
dtypes: float64(5), int64(1), object(46)
```

8. Выборка полей и строк

Для того, чтобы выбрать определенное поле, необходимо указать его имя в качестве индекса над объектом DataFrame.

Ввод:

```
complaints['Complaint Type']
```

Вывод:

```
0      Noise - Street/Sidewalk
1              Illegal Parking
2           Noise - Commercial
3              Noise - Vehicle
4                       Rodent
5           Noise - Commercial
...
111065            Illegal Parking
111066    Noise - Street/Sidewalk
111067         Noise - Commercial
111068           Blocked Driveway
Name: Complaint Type, Length: 111069, dtype: object
```

Для того чтобы получить пять первых строк набора, нужно сделать срез по данным df[:5].Также можно объединить запросы и получить первые пять значений поля.

Ввод:

```
complaints['Complaint Type'][:5]
```

Вывод:

```
0    Noise - Street/Sidewalk
1            Illegal Parking
2         Noise - Commercial
3         Noise - Vehicle
4         Rodent
Name: Complaint Type, dtype: object
```

9. Выбор нескольких полей

Для выбора нескольких полей, необходимо указать их названия, как индекс набора данных.

Ввод:

```
complaints[['Complaint Type', 'Borough']]
```

Вывод:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 111069 entries, 0 to 111068
Data columns (total 2 columns):
Complaint Type    111069  non-null values
Borough           111069  non-null values
dtypes: object(2)
```

Для того, чтобы посмотреть значения первых десяти строк, нужно сделать срез по 10 элементам.

Ввод:

```
complaints[['Complaint Type', 'Borough']][:10]
```

Вывод:

	Complaint Type	Borough
0	Noise - Street/Sidewalk	QUEENS
1	Illegal Parking	QUEENS
2	Noise - Commercial	MANHATTAN
3	Noise - Vehicle	MANHATTAN
4	Rodent	MANHATTAN
5	Noise - Commercial	QUEENS
6	Blocked Driveway	QUEENS

	Complaint Type	Borough
7	Noise - Commercial	QUEENS
8	Noise - Commercial	MANHATTAN
9	Noise - Commercial	BROOKLYN

10. Выборка строк по условию

Задача: найти самый распространенный вид жалоб из указанного набора ('311-service-requests.csv'). Для начала посчитаем у каждого вида все жалобы с помощью функции value_counts().

Ввод:

```
complaints['Complaint Type'].value_counts()
```

Вывод:

```
HEATING                    14200
GENERAL CONSTRUCTION        7471
Street Light Condition      7117
DOF Literature Request      5797
PLUMBING                    5373
...
Highway Sign - Damaged             1
Ferry Permit                       1
Trans Fat                          1
DWD                                1
Length: 165, dtype: int64
```

Для того чтобы выбрать 10 самых распространенных жалоб, сделаем срез из полученного набора.

Ввод:

```
complaint_counts = complaints['Complaint
Type'].value_counts()
complaint_counts[:10]
```

Вывод:

```
HEATING                    14200
GENERAL CONSTRUCTION        7471
Street Light Condition      7117
DOF Literature Request      5797
PLUMBING                    5373
PAINT - PLASTER             5149
Blocked Driveway            4590
NONCONST                    3998
Street Condition            3473
Illegal Parking             3343
dtype: int64
```

Выбранную информацию отразим на графике (рис. 2.10).

Ввод:

```
matplotlib inline
complaint_counts[:10].plot(kind='bar')
```

Вывод:

```
<matplotlib.axes.AxesSubplot at 0x7ba2290>
```

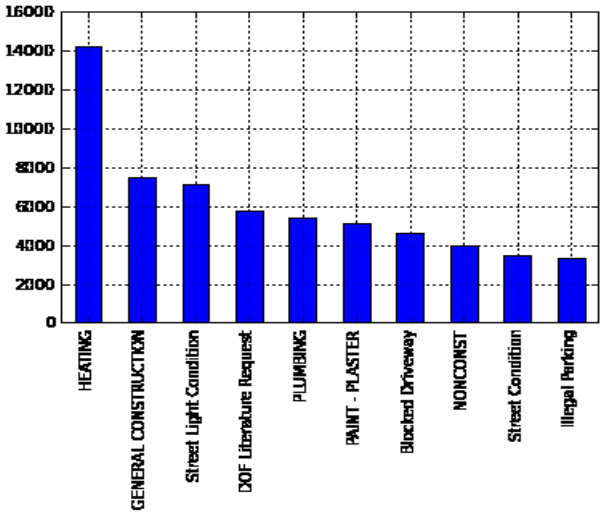

Рис. 2.10. Построенный график жалоб с отображением их количества

2.4. Лабораторная работа №4

ПАРСИНГ И АНАЛИЗ ДАННЫХ ИЗ ИНТЕРНЕТА

Цель работы: получение навыков работы по сбору информации с сайта в исходном виде, переработка полученной информации и генерация результатов проведенного анализа. Изучение функций, посредством которых осуществляется парсинг и анализ данных.

В результате выполнения лабораторных работ студенты должны знать:

- что такое парсинг и в каких целях он применяется;
- уметь анализировать полученную с сайта информацию;
- осуществлять генерацию результатов;
- знать схему действия основных функций.

Используемые программно-технические средства: персональная ЭВМ класса IBM PC AT стандартной конфигурации; операционная система Windows XP/Vista/7/8, среда разработки для Python, пакет Pandas.

Теоретическая часть

Парсинг — это синтаксический анализ информации. Под парсингом HTML, как правило, подразумевают выборочное извлечение большого количества информации с других сайтов и ее последующее использование, т.е. нахождение на странице необходимого участка кода, в котором заключена нужная информация. Если части кода повторяются в пределах одной страницы и на других страницах, имеющих аналогичную структуру, можно выделить и импортировать повторяющиеся данные автоматиче-

ски, существенно сэкономив время и предупредив возможные ошибки копирования этой информации вручную.

Для работы с данными мы будем использовать библиотеки matplotlib – это библиотека графических построений для языка программирования Python, и его расширения вычислительной математики NumPy. Библиотека Matplotlib обеспечивает объектно-ориентированный интерфейс для встраивания графиков в приложения, используя инструменты GUI общего назначения, такие как WxPython, Qt, или GTK+. Существует также процедурный pylab-интерфейс напоминающий MATLAB. Библиотека научных расчетов SciPy также использует matplotlib для работы с графиками.

Для чтения текстовых файлов используются две функции read_csv() и read_table(). Они обе используют похожий код разбора данных для преобразования табличных данных в объект DataFrame.

Под функцией "groupby" будем понимать процесс, включающий один или более шагов:

- Разделение данных на группы по каким-либо критериям;
- Применение функции к каждой группе независимо друг от друга;
- Объединение результатов в единую структуру данных.

В большинстве ситуаций вы можете разделить набор данных в группы и сделать что-то с этими группами самостоятельно. На данном этапе можно выполнить одно из следующих действий:

- Агрегация: вычисления сводной статистики (или статистики) о каждой группе. Некоторые примеры:
  - Вычисление суммы или среднего
  - Размеры группы
- Трансформация: выполнение некоторых конкретных вычислений над группой, с сохранением ее размера. Некоторые примеры:

- Стандартизация данных (zscore) в пределах группы
- Заполнение неизвестных значений (NA) в группах со значением полученного из каждой группы
    - Фильтрация: исключение групп, не удовлетворяющих поставленному условию. Некоторые примеры
    - Отбросив данные, относящиеся к группам с несколькими членами
- Фильтрация данных на основе групповой суммы или среднее значение

Некоторые сочетания GroupBy будут рассмотрены как результаты применения шагов и попытаются вернуть объединение результата, если он не вписывается ни в одну из вышеперечисленных категорий.

Функция GroupBy должна быть хорошо знакома тем, кто использовал SQL, в котором вы можете написать следующий код:

```
SELECT Column1, Column2, mean(Column3), sum(Column4)
FROMSomeTable
GROUPBYColumn1, Column2
```

Практическая часть

Поставлена задача анализа показателей температуры и осадков на велодорожках, для выделения наиболее благоприятных для поездок дней. Необходимую информацию о погоде мы возьмем с канадского сайта погоды и получим данные за март 2012 года. Для получения данных о погоде в Монреале, сохраним ссылку на сайт в переменную url_template:

```
url_template="http://climate.weather.gc.ca/climateData/
bulkdata_e.html?format=csv&stationID=5415&Year={year}&M
onth={month}&timeframe=1&submit=Download+Data"
```

Чтобы получить данные на март 2013 года, мы должны подставить соответствующие значения в строку шаблона:

```
url=url_template.format(month=3, year=2012)
```

Теперь можно загрузить данные по указанной ссылке:

```
weather_mar2012 =pd.read_csv(url, skiprows=16, in-
dex_col='Date/Time', parse_dates=True, encod-
ing='latin1')
```

Таким образом, мы передаем ссылку на данные в функцию read_csv. Параметр skiprows указывает pandas на необходимость пропустить 16 первых строк данных, в которых содержатся малозначительные для нас метаданные. Мы также указали pandas на необходимость разбора строк с датами и установили поле 'Date / Time' индексом нашего набора. В результате получится следующий набор данных:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 744 entries, 2012-03-01 00:00:00 to
2012-03-31 23:00:00
Data columns (total 24 columns):
Year                    744  non-null values
Month                   744  non-null values
Day                     744  non-null values
Time                    744  non-null values
Data Quality            744  non-null values
Temp (°C)               744  non-null values
Temp Flag               0  non-null values
Dew Point Temp (°C)     744  non-null values
Dew Point Temp Flag     0  non-null values
Rel Hum (%)             744  non-null values
Rel Hum Flag            0  non-null values
Wind Dir (10s deg)      715  non-null values
Wind Dir Flag           0  non-null values
Wind Spd (km/h)         744  non-null values
Wind Spd Flag           3  non-null values
Visibility (km)         744  non-null values
Visibility Flag         0  non-null values
Stn Press (kPa)         744  non-null values
```

```
Stn Press Flag          0  non-null values
Hmdx12  non-null values
Hmdx Flag               0  non-null values
Wind Chill              242  non-null values
Wind Chill Flag         1  non-null values
Weather                 744  non-null values
dtypes: float64(14), int64(5), object(5)
```

Изобразим данные в виде графика, используя библиотеку matplotlib (рис. 2.11).

```
weather_mar2012[u"Temp (Â°C)"].plot(figsize=(15,
5))<matplotlib.axes.AxesSubplot at 0x34e8990>
```

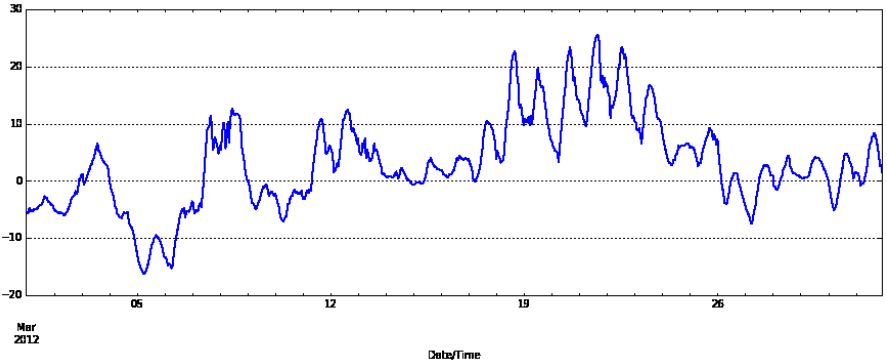

Рис. 2.11. График изменения температуры

Здесь мы использовали неверно прочитанный символ Â, для того, чтобы извлечь данные из поля с названием - Temp (°C). Чтобы удалить этот символ, вместе с символом ° из нашего набора, можно воспользоваться функцией replace над названием поля.

```
weather_mar2012.columns = [s.replace("Â°", '') for s in
weather_mar2012.columns]
```

Как вы могли заметить, есть несколько полей, которые либо полностью пустые, либо частично заполнены. От них мы можем избавиться с помощью функции dropna, которая удаляет строки или столбцы. Аргумент axis указывает, что нужно производить удаление по полям или строкам (1 и 0 соответственно), аргумент how указывает условие удаления при наличии пустых значений (any - при любом пустом значении, all - при всех пустых). Следующий код удалит те поля, в которых есть хотя бы одно пустое значение (рис. 2.12).

```
weather_mar2012 = weather_mar2012.dropna(axis=1,
how='any')
weather_mar2012[:5]
```

	Year	Month	Day	Time	Data Quality	Temp (C)	Dew Point Temp (C)	Rel Hum (%)	Wind Spd (km/h)	Visibility (km)	Stn Press (kPa)	Weather
Date/Time												
2012-03-01 00:00:00	2012	3	1	00:00		-5.5	-9.7	72	24	4.0	100.97	Snow
2012-03-01 01:00:00	2012	3	1	01:00		-5.7	-8.7	79	26	2.4	100.87	Snow
2012-03-01 02:00:00	2012	3	1	02:00		-5.4	-8.3	80	28	4.8	100.80	Snow
2012-03-01 03:00:00	2012	3	1	03:00		-4.7	-7.7	79	28	4.0	100.69	Snow
2012-03-01 04:00:00	2012	3	1	04:00		-5.4	-7.8	83	35	1.6	100.62	Snow

5 rows × 12 columns

Рис. 2.12. Таблица, где строки и столбцы, содержащие пустые значения были удалены

Данную таблицу можно оптимизировать, объединив избыточные поля Year/Month/Day/Time и столбец DataQuality, так как он не содержит интересующей нас информации. Давайте избавляться от них.

```
weather_mar2012 = weather_mar2012.drop(['Year',
'Month', 'Day', 'Time', 'Data Quality'], axis=1)
weather_mar2012[:5]
```

Вывод:

	Temp (C)	DewPoint Temp (C)	RelHum (%)	WindSpd (km/h)	Visibility (km)	StnPress (kPa)	Weather
Date/Time							
2012-03-01 00:00:00	-5.5	-9.7	72	24	4.0	100.97	Snow
2012-03-01 01:00:00	-5.7	-8.7	79	26	2.4	100.87	Snow
2012-03-01 02:00:00	-5.4	-8.3	80	28	4.8	100.80	Snow
2012-03-01 03:00:00	-4.7	-7.7	79	28	4.0	100.69	Snow
2012-03-01 04:00:00	-5.4	-7.8	83	35	1.6	100.62	Snow

Перейдем к построению температуры по времени суток (рис. 2.13). Для этого воспользуемся функциями GroupBy и aggregate.

```
temperatures= weather_mar2012[[u'Temp (C)']]
temperatures['Hour'] = weather_mar2012.index.hour
tempera-
tures.groupby('Hour').aggregate(np.median).plot()
<matplotlib.axes.AxesSubplot at 0x34ec610>
```

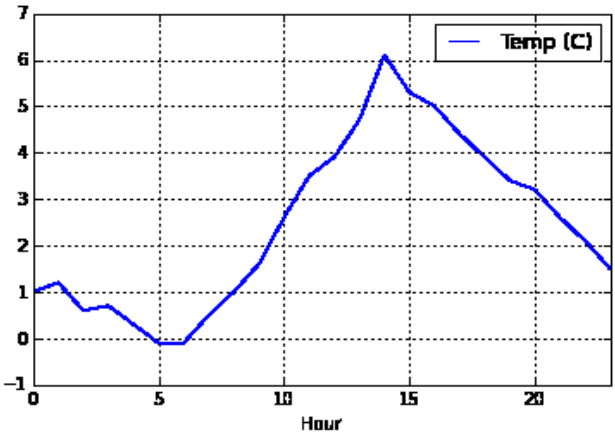

Рис. 2.13. График температуры по времени суток

Перейдем к следующему шагу - получение данных за целый год. Для начала добавим все проделанное нами ранее в отдельную функцию, которая будет возвращать погоду для данного месяца. Также, учтем ошибку, когда при выполнении запроса данных за январь, выдаются данные за предыдущий год.

```
defdownload_weather_month(year, month):
if month == 1:
year +=1
url = url_template.format(year=year, month=month)
weather_data = pd.read_csv(url, skiprows=16, in-
dex_col='Date/Time', parse_dates=True, encod-
ing="latin1")
weather_data = weather_data.dropna(axis=1)
weather_data.columns = [col.replace('Â°', '') for col
in weather_data.columns]
weather_data = weather_data.drop(['Year', 'Day',
'Month', 'Time', 'Data Quality'], axis=1)
returnweather_data
```

Можем сделать проверку работы данной функции:

```
download_weather_month(2012, 1)[:5]
```

Вывод:

	Temp (C)	DewPoint Temp (C)	RelHum (%)	WindSpd (km/h)	Visibility (km)	StnPress (kPa)	Weather
Date/ Time							
2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2012-01-01 02:00:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle, Fog
2012-01-01 03:00:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle, Fog
2012-01-01 04:00:00	-1.5	-3.3	88	7	4.8	101.23	Fog

Теперь мы сможем получить сразу данные по всем месяцам следующим образом:

```
data_by_month= [download_weather_month(2012, i) fo-
riinrange(1, 13)]
```

Теперь мы можем легко объединить все наборы данных вместе в один набор с использованием pd.concat. Получим данные за целый год.

```
weather_2012 =pd.concat(data_by_month)
weather_2012.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 744 entries, 2013-01-01 00:00:00 to
2013-01-31 23:00:00
Data columns (total 7 columns):
Temp (C)               744 non-null float64
Dew Point Temp (C)     744 non-null float64
Rel Hum (%)            744 non-null int64
Wind Spd (km/h)        744 non-null int64
Visibility (km)        744 non-null float64
Stn Press (kPa)        744 non-null float64
Weather                744 non-null object
dtypes: float64(4), int64(2), object(1)
```

Сохраним в формате CSV.

```
weather_2012.to_csv('weather_2012.csv')
```

2.5. Лабораторная работа №5

РАБОТА СО СТРОКАМИ

Целью лабораторной работы является знакомство с векторными строковыми данными в Pandas, их принципом действия; знакомство с функцией предискретизации; получение практических навыков работы со строками и данными в разных форматах, построение графиков.

В результате выполнения лабораторных работ студенты должны знать:

- принципы действия основных функций;
- схему действия функции предискретизации;
- основы работы с векторными строковыми данными и их применение;
- основные способы работы со строковыми данными в Pandas.

Используемые программно-технические средства: персональная ЭВМ класса IBM PC AT стандартной конфигурации; операционная система Windows XP/Vista/7/8, среда разработки для Python, пакет Pandas.

Теоретическая часть

1. Векторные строковые операции

Работа со строками в Pandas очень удобна за счет наличия векторных строковых операций, которые необходимы в работе со статистическими данными. Операции оснащены набором методов обработки строк, которые позволяют легко работать с каждым элементом массива. Самым главным является то, что эти методы автоматически исключают значения N/A и пропущенные значения. Они доступны через атрибуты str и, как правило, именуются в соответствии с эквивалентными встроенными строковыми методами.

Также доступны мощные методы сопоставлений с образом (pattern-matching), но стоит обратить внимание на то, что сопоставления с образом обычно используют регулярные выражения по умолчанию (а в некоторых случаях использует их всегда).

Основные функции, используемые в данной работе:

1) **pd.concat**– объединяет объекты Pandas вдоль определенной оси с дополнительным набором конфигураций на других осях. Также может добавить слой иерархической индексации на ось конкатенации, что может быть полезно, если метки одинаковы (или дублируются) на пройденной числовой оси. Аргументы:

− **axis** : *{0, 1, ...}, default 0* =None; ось для объединения;

− **dtype**; указание типа данных для данных или столбцов.

2) **resample()**– стандартный метод для преобразования и предискретизации регулярных временных рядов данных. Аргументы:

− **how**: *string*; нисходящая или повторная предискретизация, по умолчанию ‘средняя’ для нисходящей.

Практическая часть

Имеется таблица, содержащая информацию о погодных условиях за определенный период времени (рис. 2.14).

Задаем параметры отображения таблицы:

```
importpandasaspd
pd.set_option('display.mpl_style', 'default')
figsize(15, 3)
weath-
er_2012=pd.read_csv('../data/weather_2012.csv',parse_da
tes=True,index_col='Date/Time')
weather_2012[:5]
```

	Temp (C)	Dew Point Temp (C)	Rel Hum (%)	Wind Spd (km/h)	Visibility (km)	Stn Press (kPa)	Weather
Date/Time							
2012-01-01 00:00:00	-1.8	-3.9	86	4	8.0	101.24	Fog
2012-01-01 01:00:00	-1.8	-3.7	87	4	8.0	101.24	Fog
2012-01-01 02:00:00	-1.8	-3.4	89	7	4.0	101.26	Freezing Drizzle,Fog
2012-01-01 03:00:00	-1.5	-3.2	88	6	4.0	101.27	Freezing Drizzle,Fog
2012-01-01 04:00:00	-1.5	-3.3	88	7	4.8	101.23	Fog

Рис. 2.14. Таблица погоды

Можно заметить, что столбец «Погода» (Weather) содержит текстовое описание погоды за каждый час. Т.к. Pandas предоставляет возможность работы с векторными строковыми функциями, работа со столбцами, содержащими текст, становится легче. Произведем фильтрацию в столбце «Погода» по слову ‘Snow’:

```
weather_description=weather_2012['Weather']
is_snowing=weather_description.str.contains('Snow')
is_snowing[:5]
```

На выходе получим информацию в виде бинарного вектора:

```
Date/Time
2012-01-01 00:00:00    False
2012-01-01 01:00:00    False
2012-01-01 02:00:00    False
2012-01-01 03:00:00    False
2012-01-01 04:00:00    False
Name: Weather, dtype: bool
```

Для того чтобы сделать такую информацию более наглядной, построим график (рис. 2.15):

```
# Построение графика
is_snowing.plot()
```

и на выходе получим:

```
<matplotlib.axes.AxesSubplotat 0x403c190>
```

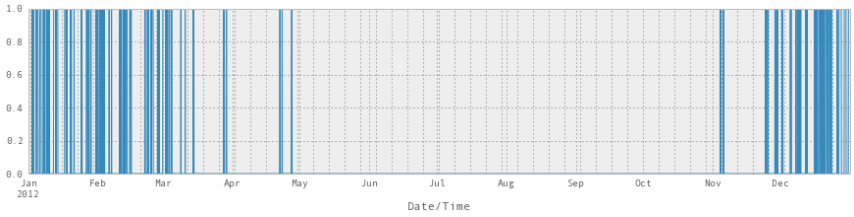

Рис. 2.15. График, отображающий, в какие часы шёл снег

Попробуем использовать предискретизацию для поиска самого снежного месяца. Использование метода resample() позволит узнать среднюю температуру для каждого месяца:

```
weather_2012['Temp
(C)'].resample('M',how=np.median).plot(kind='bar')
```

На выходе получим:

```
<matplotlib.axes.AxesSubplot at 0x560cc50>
```

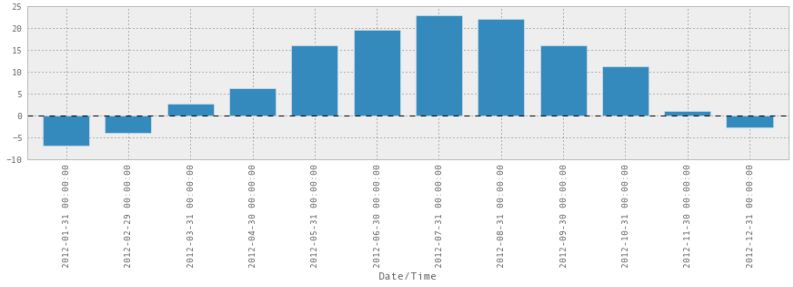

Рис. 2.16. График среднемесячных температур за год

На рис. 2.16 видим, что июль и август - самые теплые месяцы.

Представим, что наличие снегопада отображается в значениях 0 и 1, а не в True и False:

```
is_snowing.astype(float)[:10]
```

На выходе:

```
Date/Time
2012-01-01 00:00:00    0
2012-01-01 01:00:00    0
2012-01-01 02:00:00    0
2012-01-01 03:00:00    0
2012-01-01 04:00:00    0
2012-01-01 05:00:00    0
2012-01-01 06:00:00    0
2012-01-01 07:00:00    0
2012-01-01 08:00:00    0
2012-01-01 09:00:00    0
Name: Weather, dtype: float64
```

Затем применим метод resample(), чтобы найти процент времени, в которое шел снег, за каждый месяц.

```
is_snowing.astype(float).resample('M',how=np.mean)
```

На выходе:

```
Date/Time
2012-01-31    0.240591
2012-02-29    0.162356
2012-03-31    0.087366
2012-04-30    0.015278
2012-05-31    0.000000
2012-06-30    0.000000
2012-07-31    0.000000
2012-08-31    0.000000
2012-09-30    0.000000
2012-10-31    0.000000
2012-11-30    0.038889
2012-12-31    0.251344
Freq: M, dtype: float64
```

Построим график снегопадов (рис. 2.17):

```
is_snowing.astype(float).resample('M',how=np.mean).plot
(kind='bar')
<matplotlib.axes.AxesSubplot at 0x5bdedd0>
```

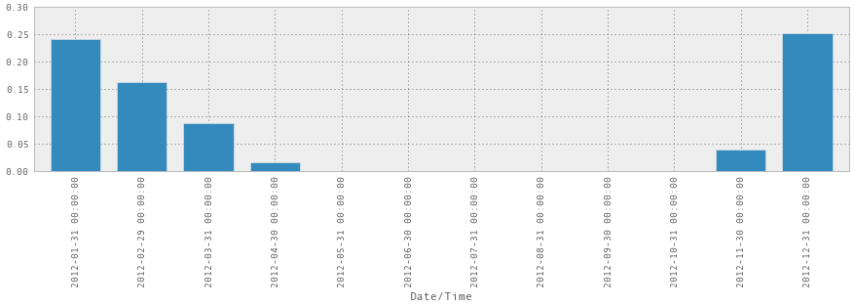

Рис. 2.17. График снегопадов

Теперь, исходя из данных, представленных в удобном виде на графике, мы можем сделать выводы о том, какой месяц был самым снежным.

Построим совместный график средних температур и снегопадов. Также можно объединить эти две статистики в один фрейм и затем построить общий график (рис. 2.18):

```
temperature=weather_2012['Temp
(C)'].resample('M',how=np.median)
is_snowing=weather_2012['Weather'].str.contains('Snow')
snowi-
ness=is_snowing.astype(float).resample('M',how=np.mean)

# Присвоимстолбцамназвания
temperature.name="Temperature"
snowiness.name="Snowiness"
```

Для объединения двух статистик в одном фрейме будем использовать функцию pd.concat.

```
stats=pd.concat([temperature,snowiness],axis=1)
stats
```

Получим:

	Temperature	Snowiness
Date/Time		
2012-01-31	-7.05	0.240591
2012-02-29	-4.10	0.162356
2012-03-31	2.60	0.087366
2012-04-30	6.30	0.015278
2012-05-31	16.05	0.000000
2012-06-30	19.60	0.000000
2012-07-31	22.90	0.000000
2012-08-31	22.20	0.000000
2012-09-30	16.10	0.000000
2012-10-31	11.30	0.000000
2012-11-30	1.05	0.038889
2012-12-31	-2.85	0.251344

Рис. 2.18. Объединенные в одну таблицу статистики снегопадов и средних температур

Построим объединенный график (рис. 2.19):

```
stats.plot(kind='bar')
```

Получим:

```
<matplotlib.axes.AxesSubplot at 0x5f59d50>
```

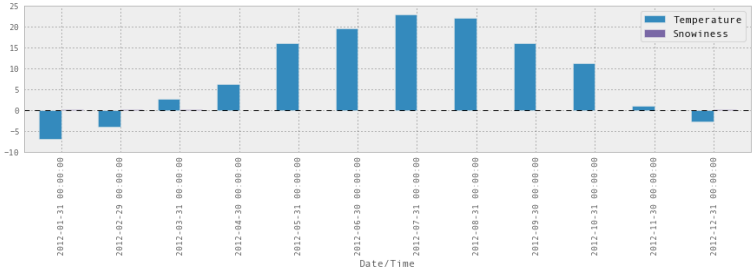

Рис. 2.19. Объединенные графики снегопадов и средних температур

Для большей наглядности можно разделить два графика и разместить их друг под другом (рис. 2.20):

```
stats.plot(kind='bar',subplots=True,figsize=(15,10))

array([<matplotlib.axes.AxesSubplot object at
0x5fbc150>,
<matplotlib.axes.AxesSubplot object at 0x60ea0d0>],
dtype=object)
```

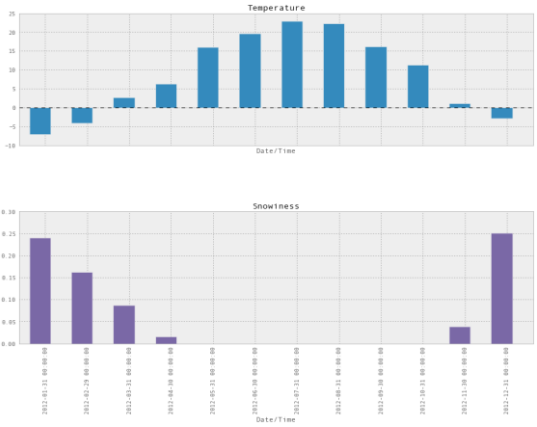

Рис. 2.20. Отформатированный вывод графиков снегопадов и средних температур